TP2 - résolution de problèmes combinatoires

Remise le 3 novembre sur Moodle.

Consignes

- Le TP peuvent être faits par groupe de 3 au maximum. Il est fortement recommandé d'être 3.
- Lors de votre soumission sur Moodle, donnez votre rapport (format pdf), votre fonction `solve` (python ou autre), et vos 4 modèles minizinc (.mzn)
- Indiquez votre nom dans le fichier pdf soumis.
- Toutes les consignes générales du cours (interdiction de plagiat, etc.) s'appliquent pour ce TP.

Conseils

Ce TP est volontairement *challenging* pour obtenir la totalité des points. Voici quelques conseils pour le mener à bien :

1. Prenez-y vous tôt. Il y a également la courbe d'apprentissage de Minizinc à considérer.
2. Travaillez efficacement en groupe, et répartissez vous bien les tâches.
3. Tirez le meilleur parti des séances de TP encadrées afin de demander des conseils.
4. Pour la partie de modélisation, identifiez quelles contraintes globales pourraient vous aider.
5. Pour la partie recherche locale, ne consacrez pas tout votre temps à l'amélioration de votre solution. Gardez cela pour la fin une fois que le reste est clôturé.

Bonne chance !

1 Résolution par recherche locale (10 pts)

En l'an 2767, sur la planète *Géonosis*, un cartel dans le domaine énergétique a construit une série de générateur d'électricité $(G_1, \ldots, G_n)$ localisé à différents endroits de la planète. Le cartel a une série de machines industrielles $(M_1, \ldots, M_m)$ qui doivent être alimentées par l'électricité fournie par les générateurs. Chaque machine doit être alimentée par exactement un générateur. Avant de pouvoir alimenter une machine, le générateur en question doit avoir été activé préalablement. Ainsi, on a deux niveaux de décision à considérer :

1. Quels générateurs va t-on activer ?
2. Quel générateur va t-on assigner à chaque machine ?

L'activation d'un générateur i a un coût fixe c_i et l'alimentation à une machine j par le générateur i a un coût variable $d_{i,j}$ qui correspond à leur distance euclidienne respective. L'objectif est de minimiser les coûts totaux engendrés sur base de nos deux niveaux de décision. Il vous est demandé de résoudre ce problème en utilisant la recherche locale.

Afin de vous aider dans votre tâche, un générateur aléatoires d'instances, un algorithme de résolution naïf (ouvrir tous les générateurs, et assigner chaque machine au générateur le plus proche), un code de visualisation de solutions, et un vérificateur partiel de solutions vous sont fournis. Il vous est demandé

de (1) résoudre au mieux ce problème en utilisant la recherche locale, et (2) fournir un court rapport (une page au maximum) expliquant votre modèle et algorithme (voisinage(s), fonction d'évaluation, fonction de sélection, heuristique, métaheuristiques, restarts, etc.). Vous avez une très grande liberté pour résoudre ce problème, vous pouvez également utiliser de algorithmes de recherche locale non-vus au cours. Les points pour cette question sont répartis comme suit :

- 4 points sur 10 seront attribués pour la qualité de votre code. Donne t-il une solution faisable meilleure que la solution naïve ?
- 3 points sur 10 seront attribués pour la qualité de votre rapport. Est-ce que votre modèle et vos choix de conception sont clairement expliqués ?
- 1 point sur 10 sera attribué pour les 75% meilleurs groupes, en terme de la qualité de la solution trouvée.
- 1 point sur 10 sera attribué pour les 50% meilleurs groupes, en terme de la qualité de la solution trouvée.
- 1 point sur 10 sera attribué pour les 25% meilleurs groupes, en terme de la qualité de la solution trouvée.

En cas d'égalité entre les groupes, les points seront toujours attribués à votre faveur. Notez également que soumettre un code qui ne compile pas, ou qui donne une solution infaisable, sera sanctionné. Votre code sera évalué sur 10 instances générées aléatoirement avec le même générateur que celui qui vous est disponible [1], avec **100 machines**, et **25 générateurs**. Ainsi, il est fortement déconseillé de *hardcoder* votre solution sur une instance spécifique. la base du code est implémentée en python et utilise que des librairies de base (`math`, `random`, `matplotlib`, `argparse`), du coup aucun environnement conda n'est fourni. Le temps d'exécution alloué pour votre algorithme de recherche sera de 5 minutes. En guise d'illustration, le schéma suivant vous montre une solution possible. Si votre algorithme comporte de l'aléatoire, 5 runs seront effectués par instance, et la meilleure solution trouvée sera considérée.

Afin de faciliter la correction il vous est demandé de compléter la fonction `solve.py` avec votre propre algorithme. Vous ne devez pas changer le fichier `generaror_problem.py`. L'output attendu est le suivant :

```
$ python main.py --n_generator=5 --n_device=10 --seed=1

(ASSIGNED-GENERATOR) [4, 0, 2, 0, 4, 1, 3, 1, 1, 0] // gen. pour chaque machine
(OPENED-GENERATOR) [1, 1, 1, 1, 1] // generateur ouvert ou non
(SOLUTION-COST) 2371.024296211267 // coût de la solution trouvée
```

1. Concrètement, on va garder le même code et juste changer la seed.

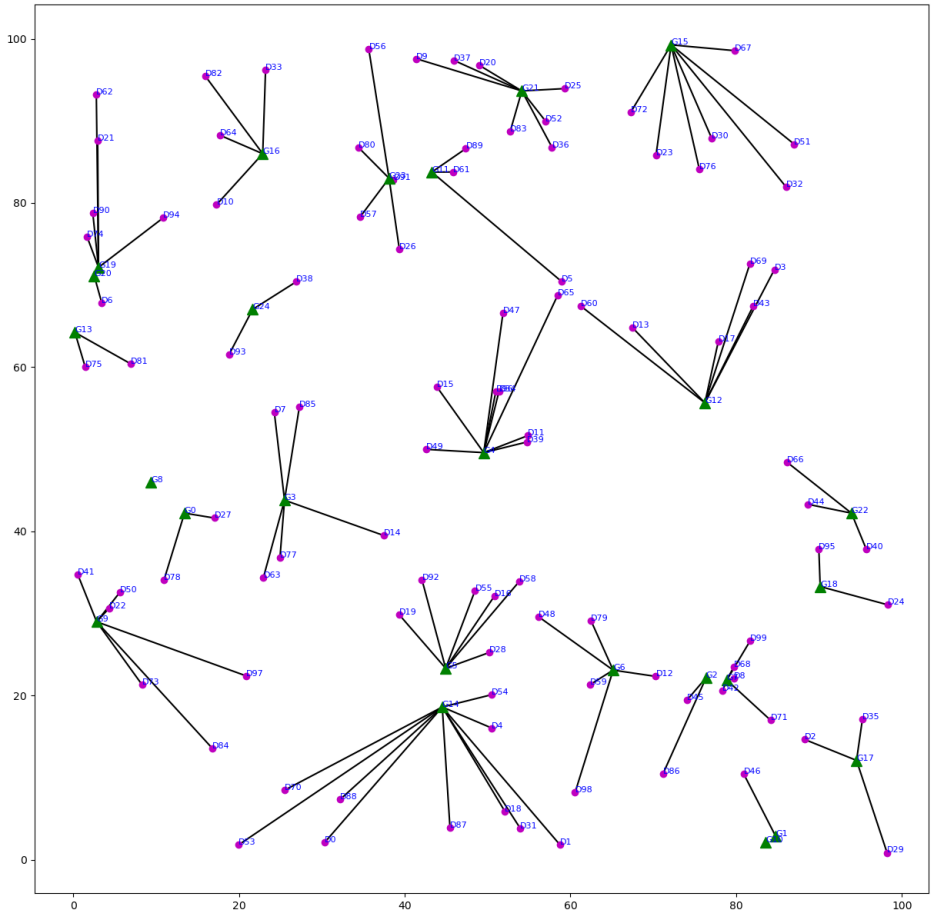


FIGURE 1 – Solution possible (100 machines, et 25 générateurs qui sont tous ouverts, génération avec une seed de 1)

2 Modélisation en programmation par contraintes (10 pts)

Pour cette deuxième partie du TP, il vous est demandé de modéliser différents problèmes, de difficulté croissante, sur MiniZinc. Pour chaque exercice, un fichier de base `.mzn`, ainsi que différentes configurations à résoudre `.dzn` vous seront fournies. Vous êtes fortement encouragé à utiliser l'API de minizinc et d'identifier les contraintes globales les plus pertinentes pour résoudre les problèmes.

Vos modèles devraient être capable de résoudre chaque configuration en moins de 5 minutes [2]. Lors de la soumission, envoyez seulement vos trois modèles Minizinc.

2. Et je compte très large, tout peut être résolu en moins de 10 secondes sur mon laptop 2,6 GHz Intel Core i5.

2.1 Drones de surveillance - easy level (0 pts)

Vous êtes en charge de la surveillance d'une zone carrée de dimension $n \times n$. Pour cela, vous disposez de n drones qui ont la faculté de détecter une intrusion sur les cases se trouvant sur les positions situées à la verticale, à l'horizontale et aux diagonales de sa position. Ces lignes définissent le champ de vision du drone. Lorsqu'un drone se situe dans le champ de vision d'un autre, il y a une interférence : ils vont détecter une intrusion alors qu'il ne s'agit que d'un autre drone. Votre objectif est de placer n drones sur la zone de sorte qu'aucun champ de vision ne se superpose.

Implémentez un modèle générique pour résoudre ce problème. Cet exercice est **non côté** et a juste pour objectif de vous familiariser avec MiniZinc. Vous ne devez pas l'intégrer dans votre soumission.

2.2 Détection de mines - normal level (4.5 pts)

Vos compétences de modélisation sur le problème de surveillance ayant impressionné votre employeur, il vous envoie sur une mission plus délicate. Vous êtes en charge de la détection de mines explosives sur un terrain miné. Soit une grille carrée de taille 10×10 partiellement remplies de mines. L'objectif est de déterminer qu'elles sont les autres cases où se trouvent les mines. Grâce à des données top-secrètes que vous avez pu récupérer, vous avez pu déceler certaines informations sur la localisation des mines.

1. Pour chaque ligne et colonne, le nombre de cases avec et sans mine doit être identique.
2. Pour chaque ligne et colonne, Il ne peut pas y avoir plus de deux cases consécutives sans mine.
3. Pour chaque ligne et colonne, Il ne peut pas y avoir plus de deux cases consécutives avec mine.
4. Il ne peut pas y avoir deux lignes identiques (i.e., ayant des mines exactement au même endroit).
5. Il ne peut pas y avoir deux colonnes identiques (i.e., ayant des mines exactement au même endroit).
6. Pour certaines cases, vous avez l'information exacte de s'il y a une mine ou non.

Par exemple dans l'exemple ci-dessous (grille 4×4), la grille de gauche montre une grille partiellement complète (situation initiale du problème), celle du milieu montre une solution possible, et celle du droite montre une solution infaisable car on a trois mines consécutives sur la même ligne.

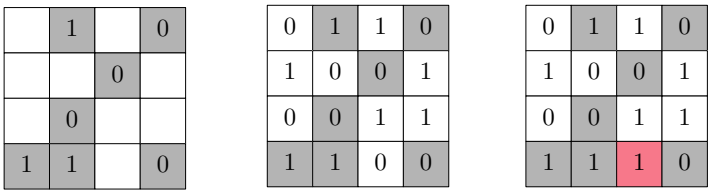


FIGURE 2 – Exemple de situation. un 1 (ou 0) donne la présence (ou non) d'une mine sur la case.

Implémentez un modèle générique pour résoudre ce problème. Trois configurations vous sont données dans des fichiers `.dzn`. Trouver une solution faisable pour chaque configuration vous rapporte 1.5 pt.

2.3 Les tours de Coruscant - nightmare level (5.5 pts)

Le gouvernement de la nouvelle république fait appel à vos compétences de modélisation pour la construction de nouvelles tours sénatoriales sur la planète Coruscant [3]. Pour ce faire, un terrain est mis à disposition. Ce dernier a 25 emplacements disponibles disposés sous la forme d'un terrain de taille 5×5. A chaque emplacement, une tour ayant une taille de 10, 20, 30, 40 ou 50 mètres doit être construite. Le sénat souhaite que l'aménagement respecte deux types de contraintes bien spécifiques :

1. Pour chaque ligne et colonne du terrain, on ne peut pas avoir deux tours ayant la même taille.
2. Le nombre de tours qu'un observateur situé sur un des bords du terrain est capable de voir est fixé à une certaine valeur. La subtilité réside dans le fait que l'observateur ne peut pas voir une tour s'il se trouve derrière une autre tour de plus grande taille suivant le champ de vision de l'observateur.

Afin d'illustrer cette seconde contrainte, considérez l'image suivante. L'observateur n'est capable que de voir trois tours (bleu, vert et orange) car le rouge et le violet sont cachés par au moins une autre tour, plus grande, se trouvant devant dans le champ de vision.

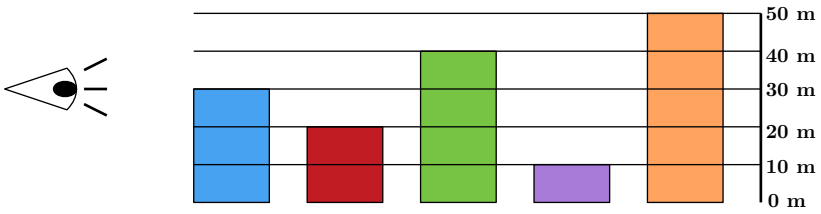


FIGURE 3 – Champ de vision d'un observateur.

La situation que vous devez résoudre est reprise sur la prochaine figure. Les cases grises correspondent au terrain (25 emplacements) et les cases vertes aux observateurs ayant leur champ de vision sur la ligne/colonne où ils se situent. Les nombres indiqués sur les cases vertes sont les données du problème et correspondent au nombre de tours que l'observateur est capable de voir.

	2	1	3	2	2	
2						3
1						4
3						1
2						3
3						2
	3	4	2	1	2	

FIGURE 4 – Situation à résoudre.

Implémentez un modèle générique pour résoudre ce problème. Une seule configuration vous est donnée. Un modèle qui trouve une solution faisable vous assure la totalité des points.

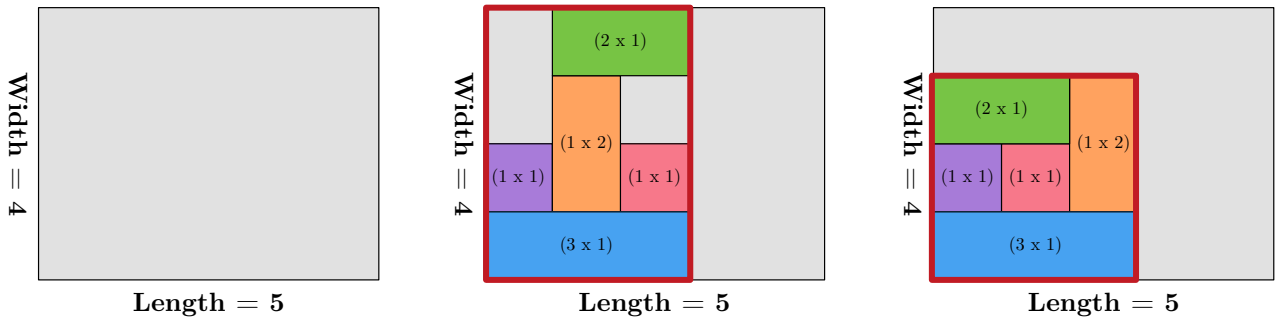
3. Rien de politique là dedans, simplement une référence à Star Wars :-)

2.4 Des vacances bien méritées... - hell level (bonus - 3 pts)

Note : cet exercice est un bonus pour les étudiants voulant se frotter à des problèmes plus compliqués. Il est ainsi non-obligatoire. La note totale du TP ne peut dépasser 20/20.

Après vos différentes missions, vous décidez de vous retirer dans votre chalet afin de profiter de vos vacances. Lors de votre arrivée, vous vous rendez compte que vous avez un grand nombre de meubles prenant énormément de place. Dès lors, votre dernière tâche avant vos vacances est de libérer cet espace.

Vous décidez d'entreposer vos meubles dans un entrepôt en attendant de pouvoir les vendre. L'entrepôt a une taille limitée et on doit pouvoir y stocker dedans un ensemble n de meubles, ayant chacun une longueur et une largeur. L'objectif est d'entreposer tous les meubles dans l'entrepôt de sorte à minimiser la *surface utile* occupée par les meubles. La *surface utile* est définie comme la surface prise par le plus petit rectangle englobant tous les meubles de l'entrepôt. Ceci est illustré dans la figure suivante. Pour simplifier le problème, on suppose que les meubles ne peuvent ni être empilés, ni être tournés.



On a un entrepôt de taille 5×4 (figure de gauche) et 5 meubles à placer, ayant chacun une taille spécifique. L'arrangement présenté sur la figure du milieu occupe une surface utile de $3 \times 4 = 12$, tandis que celui de droite occupe une surface utile de $3 \times 3 = 9$. Ce deuxième arrangement est donc préférable. Cet objectif à pour but de pouvoir minimiser les frais d'entreposage, qui sont facturés en fonction de la surface utile prise.

Implémentez un modèle générique pour résoudre ce problème. Trois configurations vous sont données dans des fichiers `.dzn`. Résoudre une configuration vous rapporte 1 pt (0.5 pour trouver au moins une solution faisable, et 0.5 pour trouver la solution optimale).

Vos missions sont maintenant achevées !